KARACRIX 入門実用ガイド

5章　簡易防犯システムの構築（基礎編）

（章別取扱説明書 v3.00）

株式会社 エスアイ創房

■改定履歴

第 3.00 版　2009/11/01

■ おことわり

(1) 本書内容の一部又は全部を、無断で他に転載することは禁止されています。

(2) 本書内容は、将来予告無く変更する場合があります。

目次

5章　簡易防犯システムの作成（基礎編）

センサとしてリードスイッチを使用した**簡単な防犯システム**を作ってみます。警報装置(アクチュエータ)としてブザーを使用しています。リモート I/O 装置には、弊社のデジタル・アナログ I/O 装置 KaracriBoard-TK0040A（以下 TK0040A）を使用します。
本システムは、DI、DO ポイント(オブジェクト)の基本的な使用法を理解して頂くことが目的です。

5.1　システム設計

それでは、どのようなシステムにするか簡単に設計しましょう。まず、どのような機能を実現するか考えます。今回は、窓やドアが強制的に開けられた時に警報を音で知らせる簡単なものにしました。家の見取り図を作ってセンサの設置場所を検討します。PC へ接続するための配線経路も考えておきます。

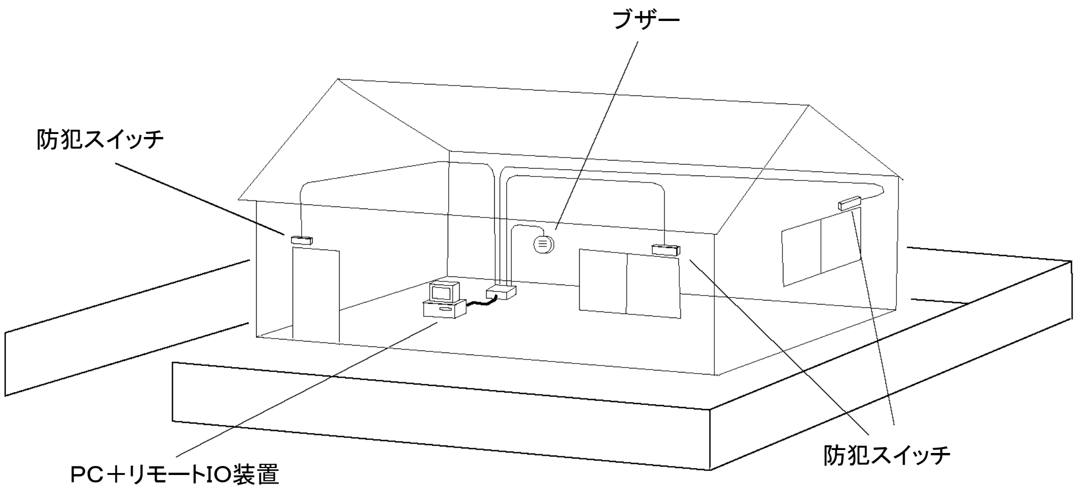


図 5.1.1 簡易防犯システムの概要

5.2　センサ、アクチュエータの設置

リードスイッチは、DIY センターなどで「防犯スイッチ」などの名称で手にいれることができます[写真 5.2.1]。リードスイッチは、磁石を離すとスイッチが ON するON型と、磁石を離すとスイッチが OFF するOFF型が市販されていますが、今回はON型を使用して窓が 開いている時 にスイッチが ON になって検知するようにしています。 リードスイッチを、窓や扉に設置して配線をリモート I/O 装置に接続します。

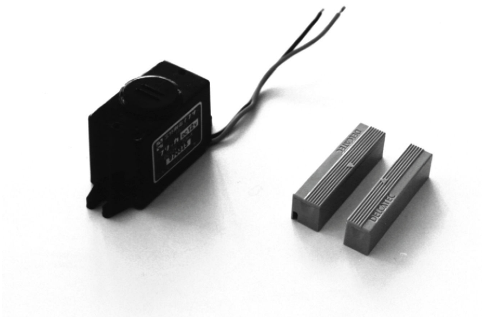


写真 5.2.1 ブザーと防犯スイッチ(リードスイッチ)

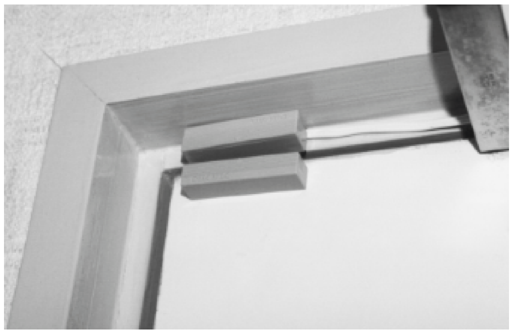


写真 5.2.2 防犯スイッチの設置例

警報装置として DC12V で動作するブザーを使用しています。電源には、TK0040A の 12V 電源端子から(AC アダプタから内部ポリスイッチ(ヒューズ)を経由して直通)の出力電源を使用します。ブザーの ON/OFF はデジタル出力(DO ポイントオブジェクト)として扱います。ブザーの消費電流が 12V で約 20mA ですので、TK0040A の内部電源とリレー(オプション装着部品)接点出力(2A)で直接駆動できます。端子台の DO-1(1c-1a)、12V、G(グランド)端子を図 5.2.1 のように接続して下さい。「防犯スイッチ」の ON/OFF 状態は、デジタル入力(DI ポイントオブジェクト)として扱います。端子台の DI-1(C-1)～DI-4(C-4)に接続して下さい。

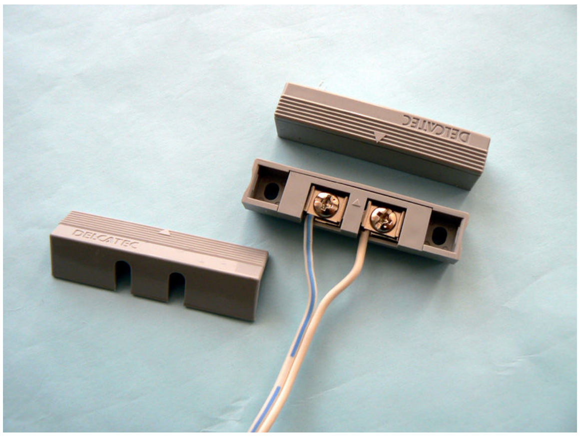


写真 5.2.3 防犯スイッチの配線

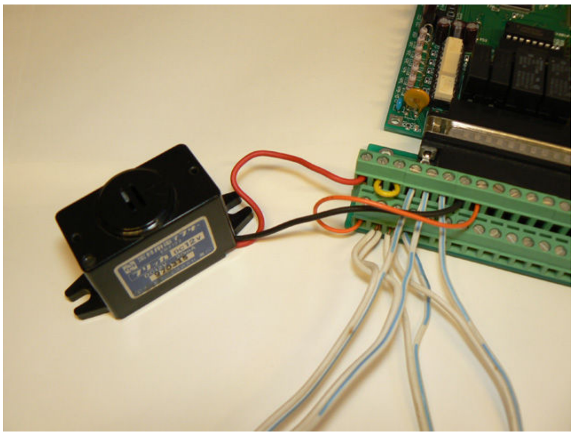


写真 5.2.4 リモート I/O 装置の配線

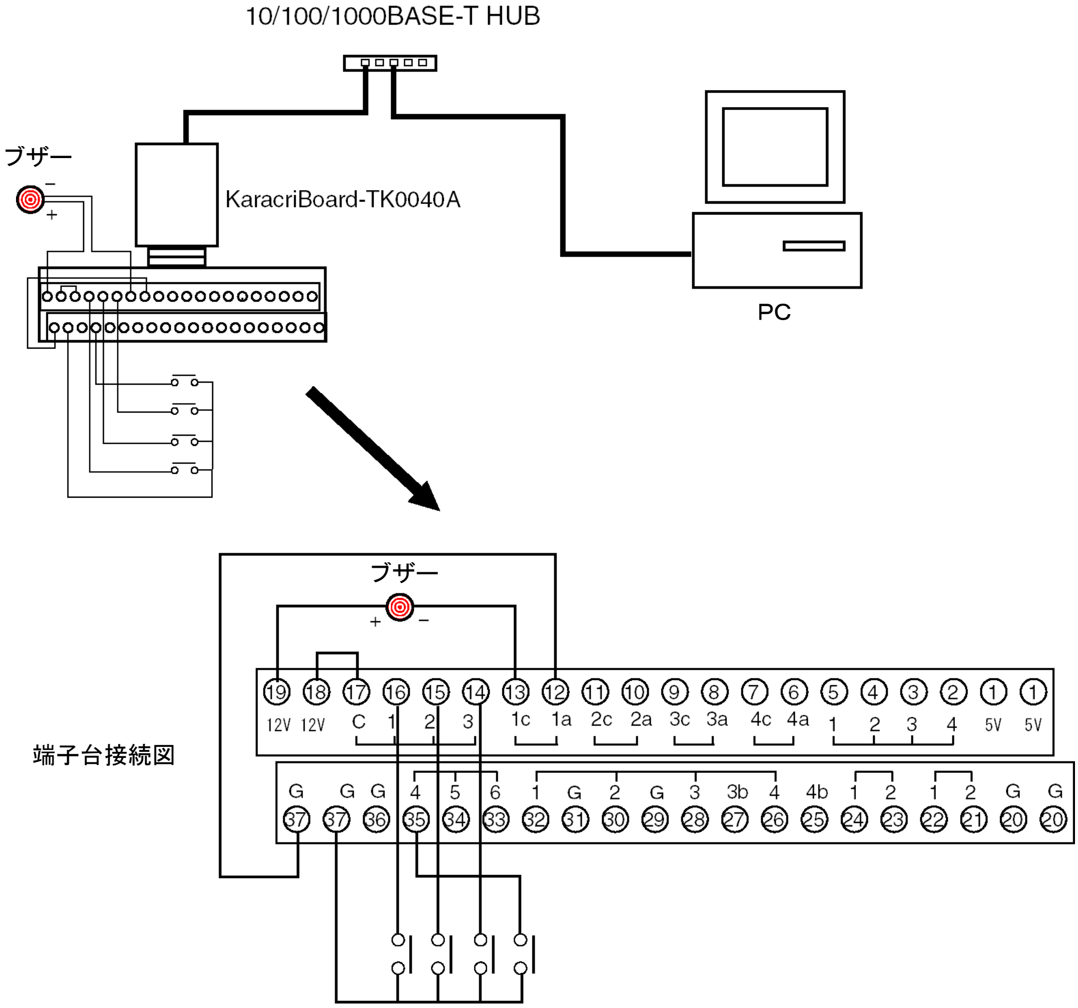


図 5.2.1 簡易防犯システム配線図

5.3　ポイント登録

KaracrixBuilder 上では、センサやアクチュエータなどの入出力対象となるものを、ポイントオブジェクトと呼んで抽象化して扱います。ここでは、「防犯スイッチ」「ブザー」をポイントオブジェクト(以下ポイント)として扱います。今回は、防犯スイッチ 4 個、ブザー1 個を使用してシステムを構築します。各ポイントには名前をつけて予め整理しておきます。

表 5.3.1 簡易防犯システムのポイント登録準備

機器	OBJID	ポイント名	ポイント種別
防犯スイッチ１	di001	玄関ドア	ＤＩ（デジタル入力）
防犯スイッチ２	di002	居間窓	ＤＩ（デジタル入力）
防犯スイッチ３	di003	和室窓	ＤＩ（デジタル入力）
防犯スイッチ４	di004	寝室窓	ＤＩ（デジタル入力）
ブザー	do001	防犯ブザー	ＤＯ（デジタル出力）

以上のように準備できたら、ポイント登録を行います。

●ポイント登録

「メインメニュー」から“システム設定”ボタンを選択して「システム環境設定メニュー」画面より「ポイント登録」画面を選択して表示します。
「防犯スイッチ」は、DI ポイントとして di001～di004 に割り当てます。
「防犯ブザー」は、DO ポイントとして do001 に割り当てます。
ポイント名のデフォルトは、“N-di001”、“N-do001”のようになっていますが、表 5.3.1 のように分かりやすいポイント名に変更します。[図 5.3.1]

(DI)

No.	種別	OBJID	タグ名	ポイント名	属性設定	W3	MB
1	DI	di001	T-di001	玄関ドア	(ON/OFF) (ALM/-)	*	*
2	DI	di002	T-di002	居間窓	(ON/OFF) (ALM/-)	*	*
3	DI	di003	T-di003	和室窓	(ON/OFF) (ALM/-)	*	*
4	DI	di004	T-di004	寝室窓	(ON/OFF) (ALM/-)	*	*

(DO)

No.	種別	OBJID	タグ名	ポイント名	属性設定	W3	MB
1	DO	do001	T-do001	防犯ブザー	(ON/OFF)	*	*

図 5.3.1 ポイント名の変更登録

●ポイント属性設定

「ポイント登録」画面の“属性設定”欄を選択して「ポイント属性設定」画面を表示して下さい。
DI ポイントとして扱われる4つの防犯スイッチの属性設定を行います。先頭の項目に“状態文字”があります。デフォルトの設定値は“ON/OFF”になっていますが、防犯スイッチの ON/OFF で検出するものは窓や扉の開閉状態ですので設定を変更してみましょう。“設定”欄を選択すると選択ダイアログが表示されますので、登録されている文字列セットから“開/閉<2>”(<2>は半角2文字分の意味)を選択して下さい。この文字列セットを選択すると、“玄関ドア”、“居間窓”などに設置した防犯スイッチが ON になったとき“開”、OFF になったときに“閉”と監視パネル(Web 画面含む)や状態一覧画面などでこのポイントの状態が表示されます。

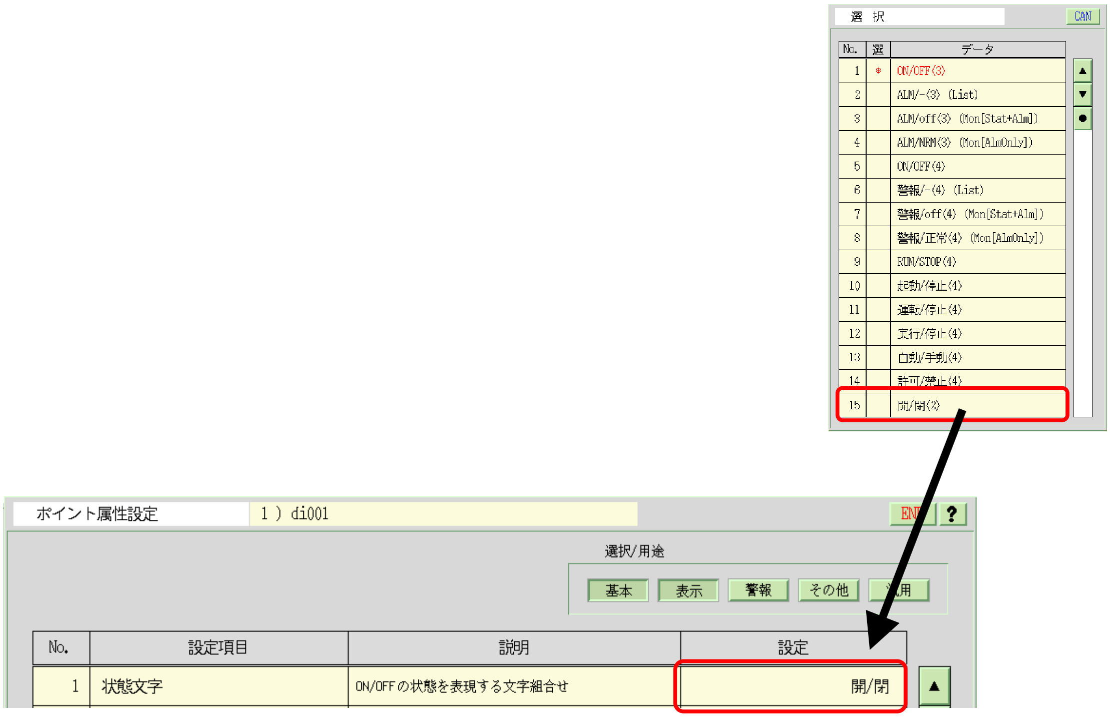


図 5.3.2 ポイント属性設定(DI)

DO ポイントとして扱われる防犯ブザーの“状態文字”は、デフォルト設定の“ON/OFF”のままでよいと思いますので、そのままの設定にしておきましょう。

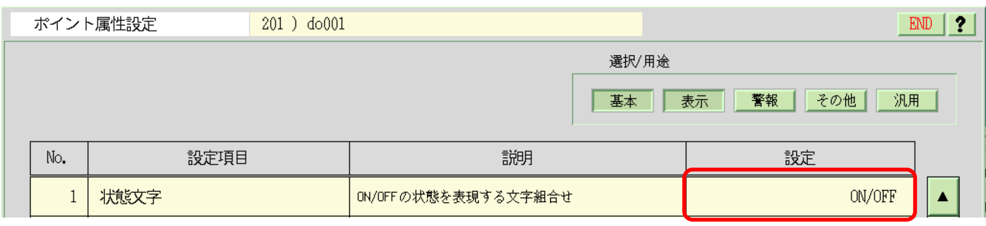


図 5.3.3 ポイント属性設定(DO)

“状態文字”の設定は、以上で終了です。この設定で、「防犯スイッチ」が ON/OFF したときの監視パネル(“状態のみ”の文字表示設定)での表示イメージ(開閉)は以下の図のようになります。



図 5.3.4 「防犯スイッチ」が ON(“開”)/OFF(“閉”)時の監視パネル表示例

<警報設定について>

ところで、属性設定には、“警報”に関連した項目があります。“警報”というのは、任意のポイントの状態が変化したときに異常値の検出として扱い、監視者に警報として通報するという考え方です。
防犯スイッチのように状態が ON になった時に危険を通報するためのポイントには、“警報”関連の属性設定を行うことで警報点として扱われます。警報点は、通常点(警報扱いしないポイント)と表現を変えて表示することが可能なため、警報状態であることを強調した表示色セットを使用して監視者に異常発生を認識させることを意図しています。

基本ボタンで選択できる項目に“警報文字”があります。警報文字の設定には、履歴用と監視画面用の2種類の設定があります。履歴用は、警報履歴一覧表示用の設定です。監視画面用は、監視パネル(Web画面含む)にグラフィック表示されるときの設定を行います。デフォルトの設定は、履歴用が“ALM/−”、監視画面用が“ALM/off”になっていますが、監視画面用の文字列セットを変更してみましょう。

2	警報文字(履歴用)	警報の有無を表現する文字組合せ	ALM/-
3	警報文字(監視画面用)	警報の有無を表現する文字組合せ	ALM/off

図 5.3.5 警報文字の設定項目

状態文字の場合と同様に選択ダイアログを表示して、登録されている文字列セットから“警報/正常<4>”(<4>は半角 4 文字分の意味)を選択して下さい。この文字列セットを選択し、文字表示設定が“警報のみ”の場合(後述)、防犯スイッチが ON になると“警報”と表示され、OFF になると“正常”が表示されます。

3	警報文字(監視画面用)	警報の有無を表現する文字組合せ	警報/正常

図 5.3.6 警報関連(監視画面用)の文字列セットを変更

次に、“通常点警報点定義”の設定値を“1”(警報点)に設定します。この値は、「通信制御ドライバ S1」で使用され、“1”に値が設定されているポイントの場合、警報を発生させるようになっています。

4	通常点警報点定義	0=通常点 1以上=警報点 (制御プログラム依存)	1

図 5.3.7 通常点警報点定義の設定

次に、“警報発生許可”の設定値が“on”になっていることを確認して下さい。この値は、「通信制御ドライバ S1」で使用され、“on”に値が設定されているポイントに警報発生を許可します。

8	警報発生許可	警報発生の実行許可 (制御プログラム依存)	on

図 5.3.8 警報発生許可の設定

そして、“警報値”の設定値を“1”に設定します。この値は、「通信制御ドライバ S1」で使用され、ポイントの値が“1”つまりデジタル接点入力の状態が閉(ショート)の ON の時に警報を発生させます。

10	警報値	警報判断用[通常1] (制御プログラム依存)	1

図 5.3.9 警報値の設定

ここで、以上の設定を取り合えずシステムに保存反映させる場合には、「ポイント登録」画面で“END”ボタンを選択して「メインメニュー」へ戻り、KaracrixBuilder コンソールの “RST”ボタンを選択して KaracrixBuilder をリセットして下さい。しばらくすると KaracrixBuilder が自動的に再起動されます。

<監視パネルにおける状態文字と警報文字の表示の使い分けについて>

監視パネルにおけるポイントの状態を表す文字は、監視パネル CAD の文字(動部品)属性の登録により表示方法が変るので説明します。

なお、以下の説明に当たり、ここまで(前項)の設定と異なりますが、
状態文字に、状態 ON のとき“ON”、OFF のとき“OFF”が設定してあり、
警報文字に、状態 ON のとき“警報”、OFF のとき“正常”が設定してあるものとします。

以下の図では、監視パネル CAD で動部品(文字)を登録している場合の例ですが、ポイント di001 の「2.表示属性」に、“状態＋警報”を指定しているところです。

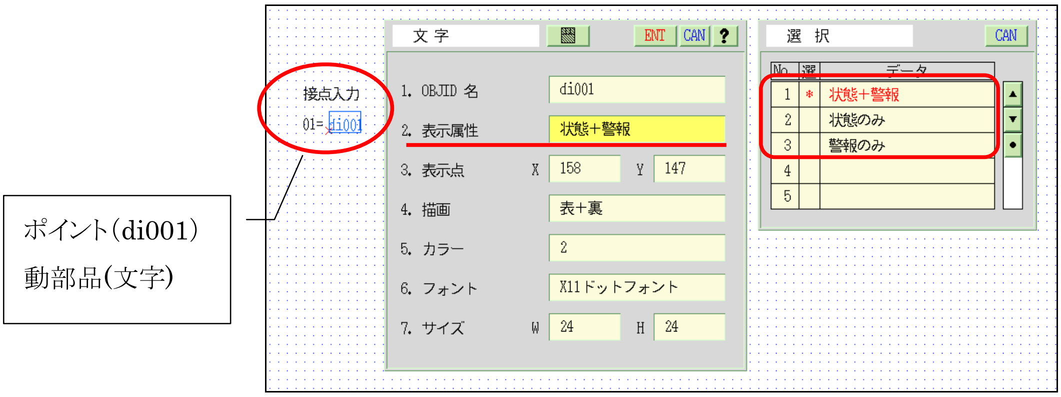


図 5.3.10 監視パネル CAD の動部品(文字)の表示属性の設定

ポイントの状態文字と警報文字は以下の3つの条件によって表示方法が異なります。

①「警報発生許可」の設定値
②ポイントの状態
③警報の発生状態

●表示属性に **状態＋警報** を設定した場合の表示は、以下の通りです。

監視パネル上では、この選択された文字列にカラーセットが組み合わされて以下のように表示されます。
※本表示属性は、警報発生許可をプログラムで動的制御して用いる場合の表示に使われます。

1. 警報発生許可が off の場合

   状態表示用の文字列セット(ON/OFF)が使用されて、状態が ON のとき“ON”、OFF のとき“OFF”になります。



図 5.3.11 「防犯スイッチ」が ON/OFF 時の監視パネル表示例 (状態＋警報:許可 OFF)

2. 警報発生許可が on の場合

状態が ON の場合には警報表示用の文字列セット(警報/正常)が使用されて、“警報”の文字が表示されます。OFF の場合には状態表示用の文字列セット(ON/OFF)が使用されて、“OFF”の文字が表示されます。



図 5.3.12「防犯スイッチ」が ON/OFF 時の監視パネル表示例（状態＋警報:許可 ON）

●表示属性に<u>状態のみ</u>を設定した場合

警報発生許可の on/off に関係なく、状態表示用の文字列セットとそのカラーセットで表示されます。状態表示用の文字列セット(ON/OFF)が使用されて、状態が ON のとき“ON”、OFF のとき“OFF”になります。



図 5.3.13「防犯スイッチ」が ON/OFF 時の監視パネル表示例（状態のみ）

●表示属性に<u>警報のみ</u>を設定した場合

状態のON/OFFについて警報表示用の文字列セットとそのカラーセットが使用されます。

1. 警報発生許可が off の場合

警報発生許可が OFF の場合は常に無警報状態と判断され、状態の ON/OFF に関係なく警報状態は OFF で、常に“正常”の文字が表示されます。



図 5.3.14「防犯スイッチ」が ON/OFF 時の監視パネル表示例（警報のみ:許可 OFF）

2. 警報発生許可が on の場合

警報表示用の文字列セット(警報/正常)が使用されて、状態が ON のとき“警報”、OFF のとき“正常”になります。



図 5.3.15「防犯スイッチ」が ON/OFF 時の監視パネル表示例（警報のみ:許可 ON）

5.4　監視パネルの作成

監視パネルは、遠方にある監視制御対象を PC 画面や Web 画面上に線画やイメージで表現して対応させるものです。PC など画面上で制御対象を遠方操作したり、監視対象の状況を知ることができます。つまり、実際の監視制御対象の相似型を PC 画面上に作成することになります。でも、現実をそっくり表現する必要はありません、要は監視制御をするあなたが十分理解できる表現で作成すればよいわけです。現実を精密に再現するよりも、デフォルメした方が理解しやすい場合も多いはずです。以上のようなことを頭に入れて、監視パネルの下絵を作成しておきます。

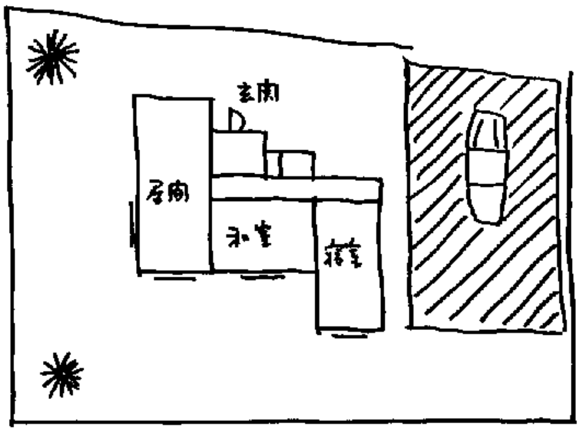


図 5.4.1 ラフスケッチを描く

それでは、スケッチに基づいての監視パネルを作成しましょう。「メインメニュー」から“監視/手動操作”ボタンを選択して「監視パネルメニュー」の空いている欄に画面名称を登録して「監視パネル」を呼び出し、「監視パネル CAD」を起動します。

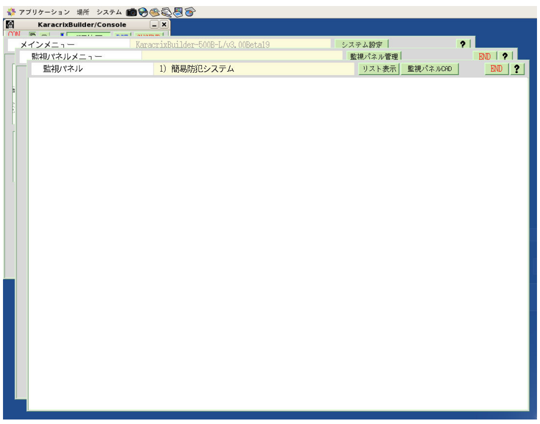


図 5.4.2 監視パネルの登録

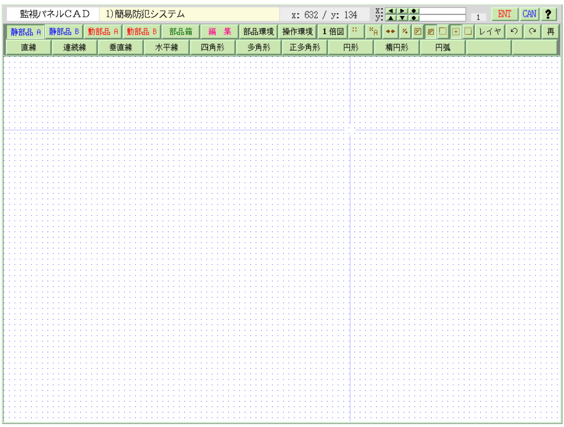


図 5.4.3 監視パネル CAD の起動

●住宅平面図を描く

[静部品 A]-[連続線] 等を選択して敷地境界の塀を描きます[図 5.4.4] 。描く線の属性はダイアログ画面内の【線属性情報】で編集します。なおデフォルト値は、予め[部品環境]-[線]で設定しておきます。

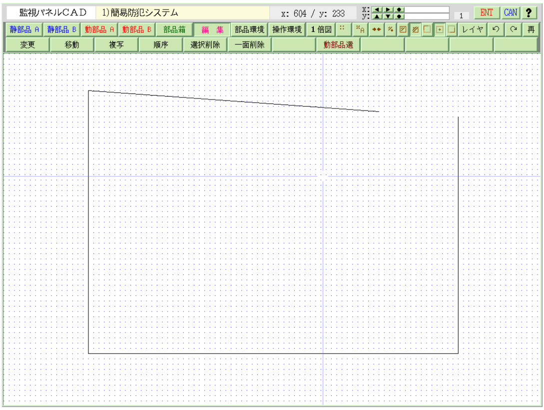


図 5.4.4 敷地境界を作図

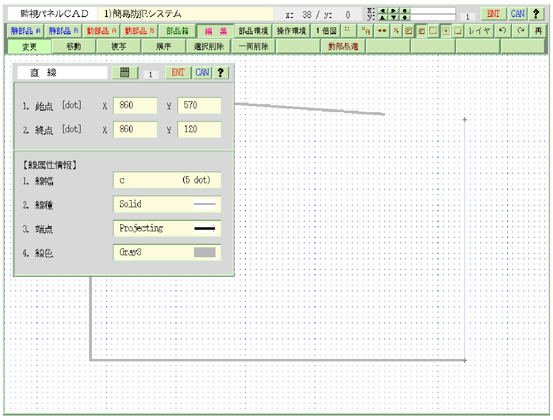


図 5.4.5 線属性の変更

次に、住宅の各部屋を描きましょう。[静部品 A]-[四角形] を選択します。ところでその前に、[丸め]ボタンを選択あるいは[操作環境]-[グリッド]で「丸める」にしておくと四角形をきれいに並べて書くことが出来ます。四角形などを使って各部屋を描画します[図 5.4.6]。
玄関の扉は、[静部品 A]-[円弧] と [静部品 A]-[直線] を組み合わせて描画します [図 5.4.7]。

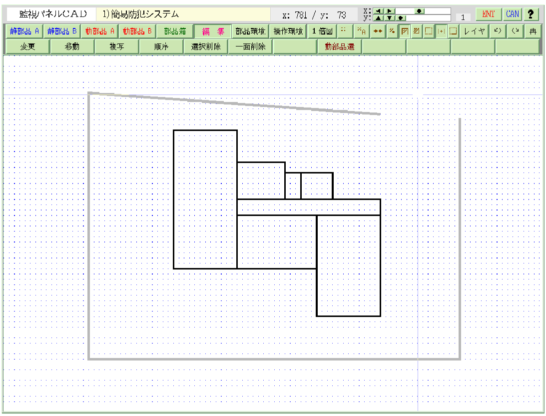


図 5.4.6 各部屋の作図

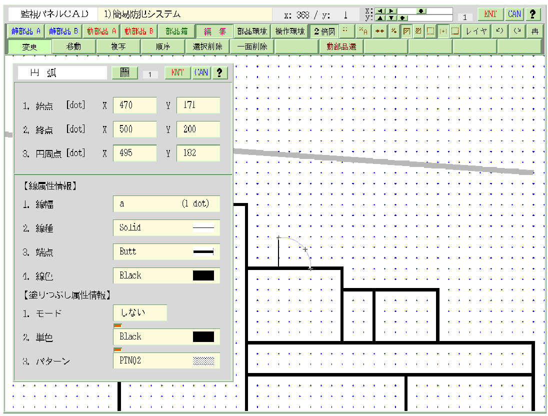


図 5.4.7 扉の作図

ガレージは、[静部品 A]-[多角形]で描画したのちに、[編集]-[変更]で ″塗りつぶし属性情報″ をパターン(モード)に変更して ″煉瓦パターン″ らしく表現しています[図 5.4.8]。
植栽は、サンプルで登録されているビットマップ部品を使用して描いています。[静部品 B]-[絵部品]を選択して、植栽のビットマップ部品を選択します[図 5.4.9]。

部屋名称は、[静部品 B]-[文字] で描画します[図 5.4.10]。

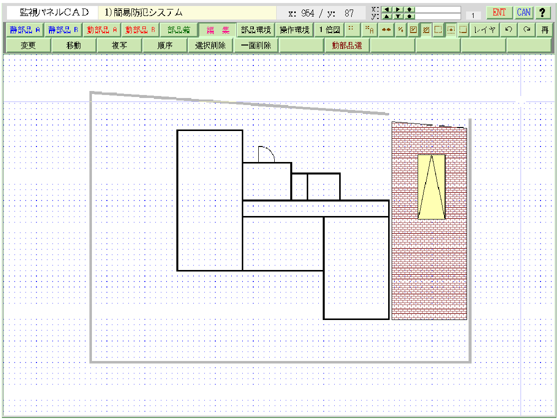


図 5.4.8 ガレージの作図

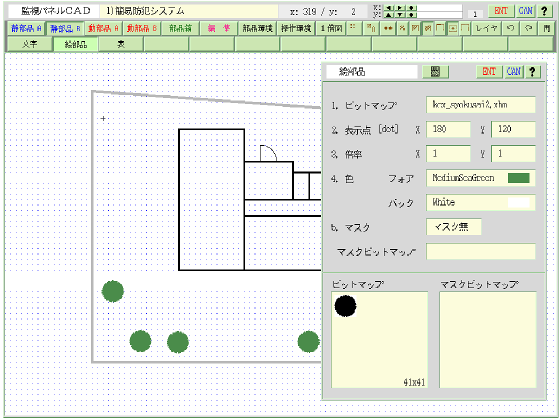


図 5.4.9 植栽の作図

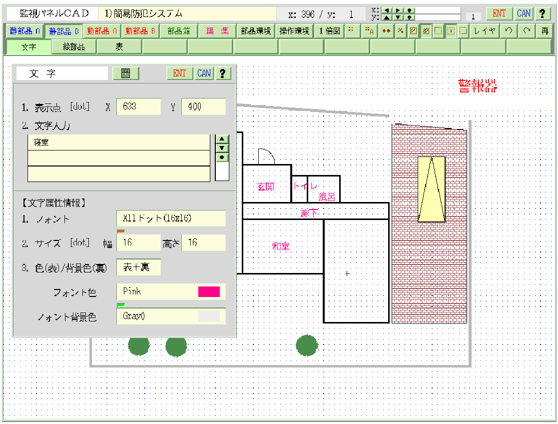


図 5.4.10 名称を入れる

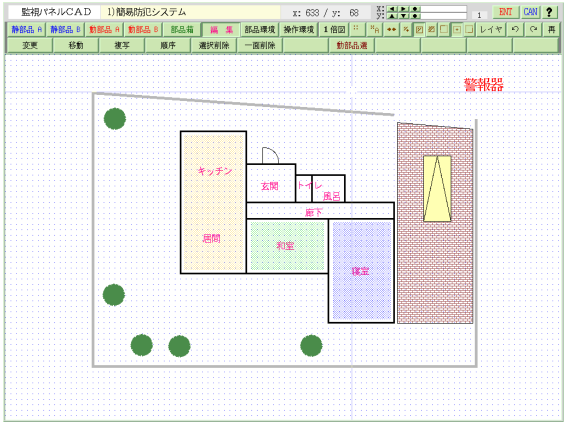


図 5.4.11 背景図の完成

背景としての住宅平面図が作成できましたら、一度"ENT"ボタンを選択して保存すると共に、監視パネルで表示されるか確認しておきます[図 5.4.11,図 5.4.12]。("CAN"ボタンは保存キャンセルとなるので注意して下さい)
監視パネルで表示されることが確認できたら、再度"監視パネル CAD"ボタンを選択して下さい。

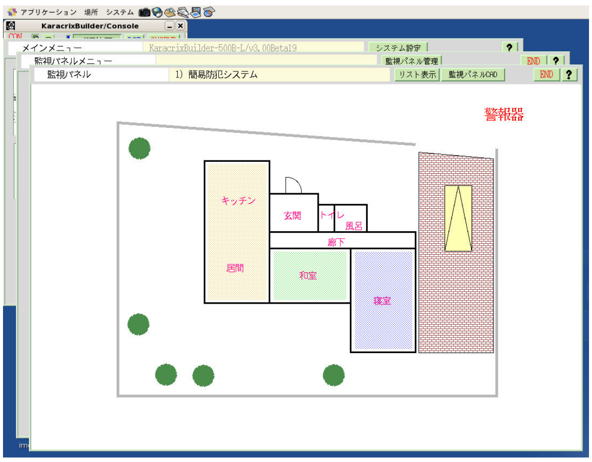


図 5.4.12 監視パネルで表示を確認

●動部品の設定

背景の住宅平面図が完成しましたら、その上にセンサ、アクチュエータとひも付けられる動部品を配置します。

下絵にある窓や扉に設置したセンサの位置に各種動部品を設定します。まず玄関の扉を“動部品 A”メニューの“直線”で設定しましょう。“直線”部品は、スイッチの状態が ON/OFF で切り替わる時に色の変化で状態を表現する部品です。[動部品 A]−[直線] を選択して下さい。

“OBJID 名”を選択して「ポイント選択」画面から OBJID を選択します。「ポイント登録」で登録した名称を選択します[図 5.4.13]。“始点”、“終点”で部品を配置する座標を入力します。“線幅”は、視認性を考えて設定して下さい。“カラー”は、DI 用のデフォルト設定(No.2/文字、線、塗面用)を使用します[図 5.4.14]。

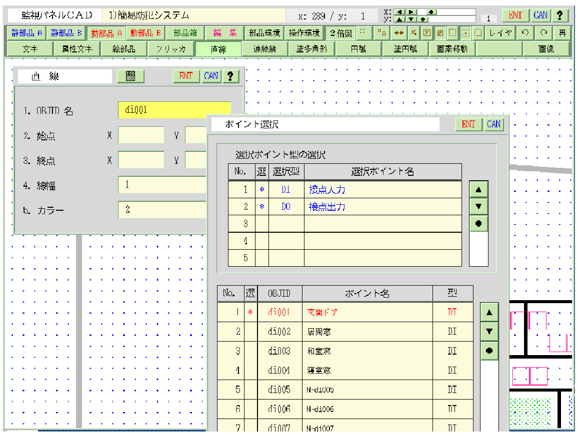


図 5.4.13 オブジェクトIDの選択

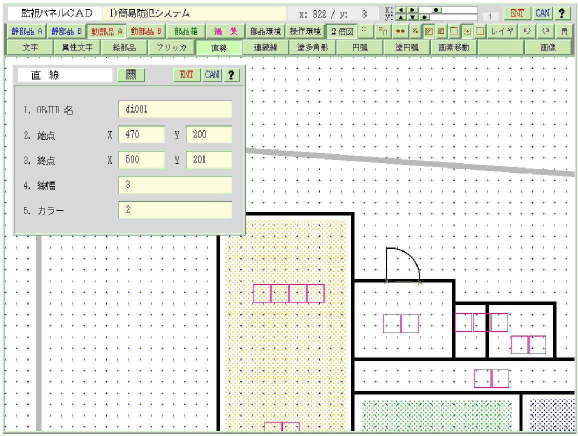


図 5.4.14 直線動部品の設定

さらに、文字を使って扉の開閉状況を表示する“文字”部品を使ってみましょう。“文字”部品は、スイッチの状態が ON/OFF で切り替わる時に表示文字列の変化で状態を表現する部品です。[動部品 A]−[文字]を選択して下さい。

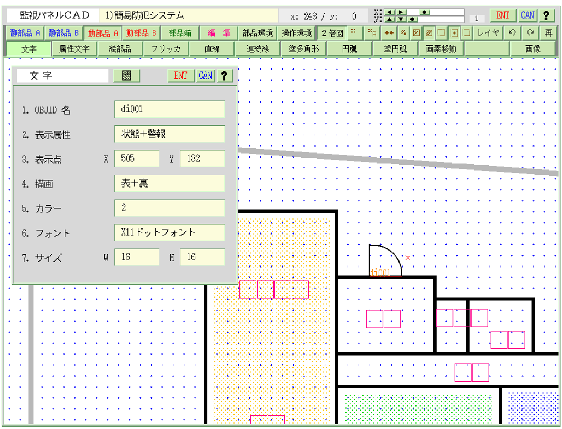


図 5.4.15 フォントの選択

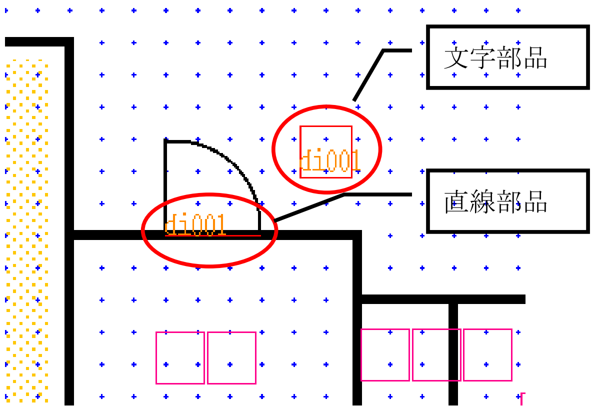


図 5.4.16 文字動部品の設定

“OBJID 名”、“表示点”、“カラー”は、“直線”動部品と同様に入力します。

“表示属性”は、“状態＋警報”を選択しています。この設定では状態文字と警報文字を交ぜて使用することが出来ます。

“描画”は、文字の背景を表示するかどうかを決めるものです。ここでは、“表＋裏”を選択して背景色も表示するように設定しています。

“フォント”の型は、X11 ドットフォント固定になっており変更できません。

“サイズ”は、OS に内蔵されているフォントのサイズから選択できます。[図 5.4.15,図 5.4.16]

そして、センサの設置されている場所に表示点を移し動部品を配置して下さい。[図 5.4.17]

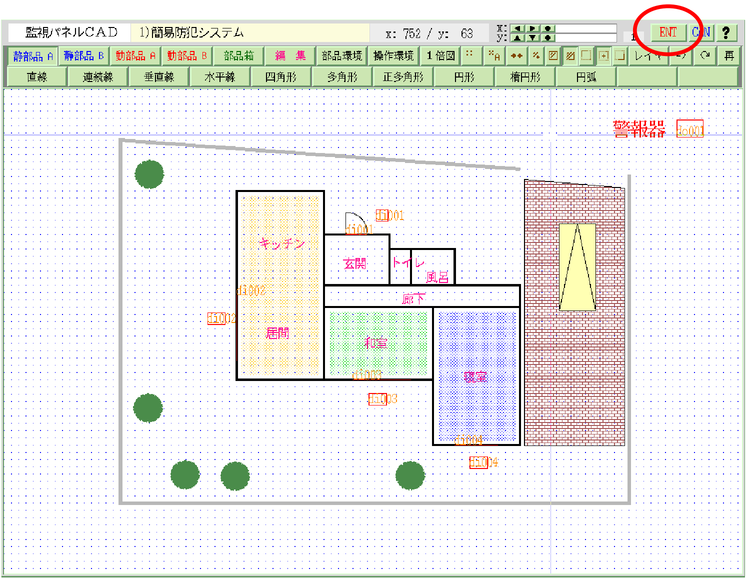


図 5.4.17 すべての動部品を設置したところ

動部品の設置ができましたら、“ENT”ボタンを選択して保存すると共に、「監視パネル」で問題なく表示されることを確認して下さい。登録部品が図の様に表示されていれば完成です[図 5.4.18]。

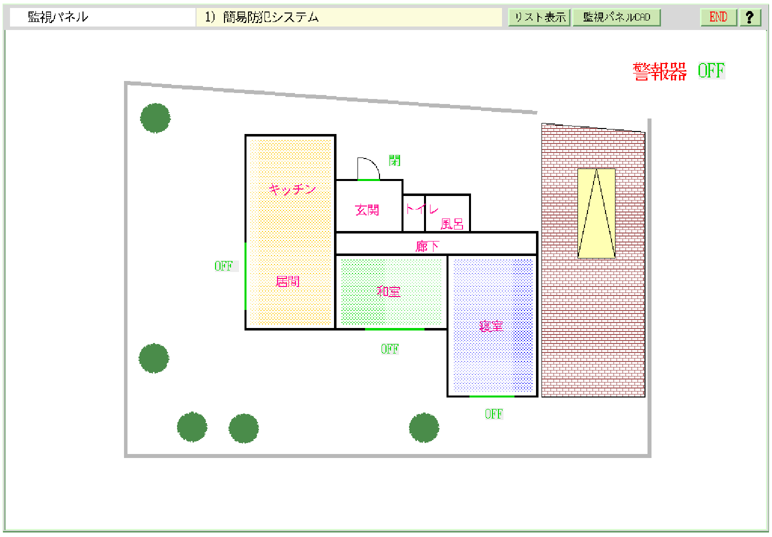


図 5.4.18 完成した監視パネル

5.5 監視パネルを使う

それでは、監視パネルでセンサやアクチュエータの状態をモニタしてみましょう。センサやアクチュエータとリモート I/O 装置の配線を確認して下さい。 また、PC とリモート I/O 装置が、正しくネットワーク接続されている事も確認します。
次に、「メインメニュー」から“制御プログラム”ボタンを選択して「制御プログラム登録」画面から“通信制御ドライバ S1”を実行して下さい。“END” ボタンで「メインメニュー」に戻ります。

次に、「メインメニュー」から“監視/手動操作”ボタンを選択して「監視パネルメニュー」から作成した監視パネルを表示して下さい。監視パネルで、玄関ドアに設定した“文字”動部品を選択してみましょう。図 5.5.1 のように状態表示ダイアログが表示されます。
状態表示ダイアログには、“ポイント名”、“OBJID”、“現在の状態”、“設定値”などの情報が表示されます[図 5.5.2]。

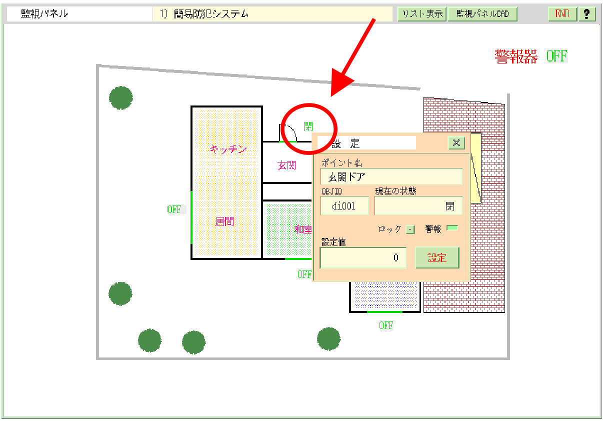


図 5.5.1 防犯スイッチの状態表示

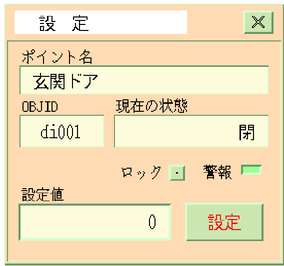


図 5.5.2 状態表示ダイアログ

次に、“防犯ブザー”を選択してみましょう。操作ダイアログが表示されます[図 5.5.3]。 ここで、“現在の状態”が“OFF”になっている場合、“ON”ボタンを押してみます。ブザーへ電源が供給されて警報が鳴れば正常に動作しています。

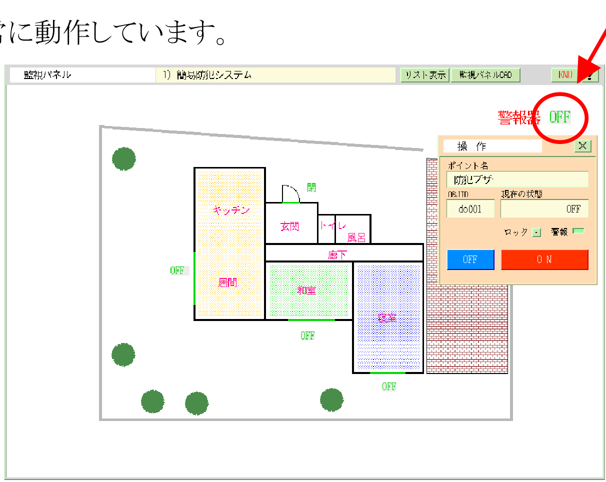


図 5.5.3 ブザーのリモート操作



図 5.5.4 操作ダイアログ

ドアの開閉をしてみて下さい。監視パネルの玄関ドアに設定した“直線”動部品に色変化がおきますか？“文字”動部品の表示が“ON”、“OFF”と変化すればこれも正常です。この状態で、ドアや窓の開閉状態は、監視することができるようになりました。

しかし、いつも監視画面の前で眺めているわけにはいきません。侵入者が窓を開けたりドアを開けたときに、自動的に警報を通知してくれるようでないと防犯システムとしては意味がありません。そこで、次の項で自動監視制御プログラムを作成してみます。

5.6　自動監視制御プログラムの作成

プログラムのアルゴリズムを考えましょう。 まず、防犯スイッチ1～4の ON/OFF 状態を常時監視する必要があります。 防犯スイッチ1～4のどれかが、ON になったら侵入者があったと解釈してブザー用のリレーを ON します。ブザーを止めるのは、監視パネルからブザーポイントオブジェクトを直接操作をして手動で止める事にします。（手動停止はサンプルプログラムを簡潔にし説明を簡単にする為です）
以上のアルゴリズムを、フロー図にすると図 5.6.1 のようになります。

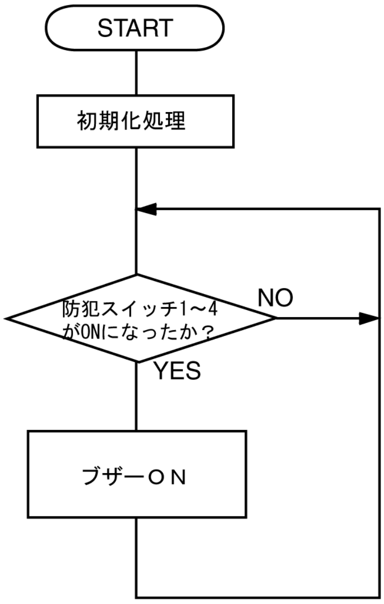


図 5.6.1 フロー図

それでは、フロー図をもとにプログラミングをしてみましょう。「メインメニュー」から “制御プログラム”ボタンを選択して「制御プログラム登録」画面を表示して下さい。
プログラムの編集方法については、外部エディタ（gedit、vi、emacs 等）を使用して作成したプログラムを KaracrixBuilder のインポート機能を使用して取り込む方法と、KaracrixBuilder の簡易編集エディタで記述する方法がありますが、詳しくは「KaracrixBuilderV3 システムマニュアル（12 章 制御プログラム）」で解説していますので、そちらを参照して下さい。
ここでは、目的の機能を実現する制御プログラムを作成する際の共通する KCX ライブラリの使用方法やポイントオブジェクトの扱い方について順を追って解説します。プログラムの編集作業自体は、外部エディタまたは、KaracrixBuilder のプログラム簡易編集画面のいずれかの方法で作成して頂くものとします。
「制御プログラム登録」画面の未登録行に“プログラム名”を登録して下さい。ここで、外部エディタで作成したプログラムを取り込む場合は、インポートの手順でプログラムを登録する手順へ進みます。また、インポートをしない場合には、プログラム名を登録した後、“プログラム編集”欄を選択すると「プログラム簡易編集」画面が表示されプログラムの編集を行うことができます。

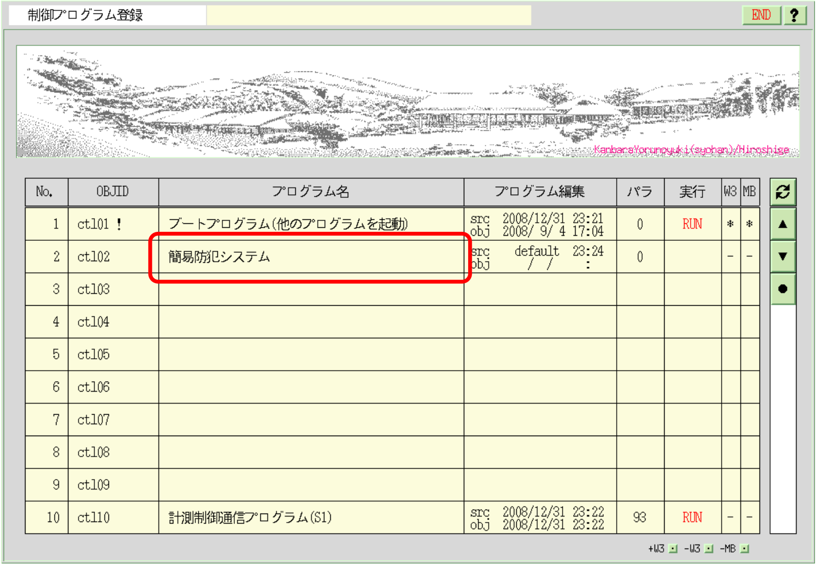


図 5.6.2 制御プログラム登録

●制御プログラム作成の手順

KaracrixBuilder でのプログラム作成の基本的な書き順は、次の様になります。

1. KCX ライブラリの初期化
2. 使用するポイントオブジェクトのオープン
3. ユーザーアルゴリズムの記述

では，順番に好みのエディタを使用するなどして記述していきましょう。

1. KCX ライブラリの初期化

kcxinit()関数は、KCX ライブラリの初期化を行います。

KaracrixBuilder でプログラムを作成する時には、main()関数の先頭で必ず実行する必要があります。

```
#include <karacrix.h>

main( int argc, char *argv[] )
{

    /* KCX ライブラリの初期化(先頭の実行文に必須) */
    kcxinit( argc, argv );

}
```

2. 使用するポイントオブジェクトのオープン

次に、kcxobj_open()関数で監視パネルに配置したポイントオブジェクトの ID 名から、KCX ライブラリ関数で使用するオブジェクト ID 識別子(整数値:ディスクリプタ)を取得します。
また、ユーザプログラム毎に必要となる各種変数の宣言等も行っておきます。
プログラムを追加していきましょう。
3 行目に、防犯スイッチの数を定義しておきます。
8-14 行目で、プログラム制御に必要となる変数の宣言をしています。
20-24 行目で、リモートI/O 装置に接続されているセンサやアクチュエータのポイントオブジェクトのオブジェクト ID 識別子を取得しています。 KCX ライブラリでは、このオブジェクト ID 識別子を使用してポイントオブジェクトの指定を記述していきます。

```
 1  #include <karacrix.h>
 2
 3  #define  SWOBJECTs   (4)   /* スイッチ 4 つを格納する配列の定義 */
 4
 5  main( int argc, char *argv[] )
 6  {
 7
 8    int   i;
 9    int   idata;                    /* 整数値を格納するバッファメモリ */
10    int   bz_flag;                  /* ブザーを鳴らすかどうかのフラグ */
11    int   bz_objid;                 /* ブザーのオブジェクト ID       */
12    int   sw_objid   [SWOBJECTs];   /* スイッチのオブジェクト ID      */
13    int   sw_crntdata[SWOBJECTs];   /* スイッチの現在の状態を格納     */
14    int   sw_savedata[SWOBJECTs];   /* スイッチの前回の状態を格納     */
15
16      /* KCX ライブラリの初期化(先頭の実行文に必須) */
17      kcxinit( argc, argv );
18
19      /* オブジェクトＩＤの取得 */
20      sw_objid[0] = kcxobj_open( ″di001″ );  /* 防犯スイッチ 1 */
21      sw_objid[1] = kcxobj_open( ″di002″ );  /* 防犯スイッチ 2 */
22      sw_objid[2] = kcxobj_open( ″di003″ );  /* 防犯スイッチ 3 */
23      sw_objid[3] = kcxobj_open( ″di004″ );  /* 防犯スイッチ 4 */
24      bz_objid    = kcxobj_open( ″do001″ );  /* ブザー         */
```

[ライブラリ関数]

kcxobj_open() 関数 ... オブジェクト ID 名からオブジェクト ID 識別子を取得

1) kcxobj_open(objidname)

char *objidname ..... オブジェクト ID 名

26-65 行目で、防犯システムのアルゴリズムの記述をしています。
kcxobj_stat_ird() 関数で、防犯スイッチの状態値を全て取得し、いずれかの値が(1)[ON状態]ならば 58 行目の kcxobj_sndistat_tokcx() 関数で、ブザーへON信号(1)を出力します。
63 行目で、sleep(システム関数)が登場していますが、これはCPUタイムを独占しないように他の忙しいプロセスなどへ制御を渡すためです。 詳しくは、OS の解説書等参照下さい。ここでは、1 秒間休止させています。

```
26      /* 初期設定 */
27      /* 防犯スイッチの状態を検定する為の初期状態を取得しておく */
28      for( i = 0; i < SWOBJECTs; i++ ){
29        kcxobj_stat_ird( sw_objid[i], &idata );
30        sw_savedata[i] = idata;
31      }
32
33      /* 防犯スイッチ 1～4 の状態を毎秒(下記 sleep(1)関数値依存)監視する永久ループ */
34      while( 1 ){
35
36        /* 防犯スイッチ 1～4 の現在の状態を取得 */
37        for( i = 0; i < SWOBJECTs; i++ ){
38          kcxobj_stat_ird( sw_objid[i], &sw_crntdata[i] );
39        }
40
41        /* 防犯スイッチ 1～4 のどこかで、ON になった時に限り          */
42        /* つまり、前回の状態が 0 で、かつ現状態が 1 になった時に限り */
43        /* ブザーフラグを ON とする                                   */
44
45        bz_flag =  0;
46
47        for( i = 0; i < SWOBJECTs; i++ ){
48          if(( sw_crntdata[i] == (1) ) &&
49             ( sw_savedata[i] == (0) ) ){
50            bz_flag =  1;
51          }
52          sw_savedata[i] =  sw_crntdata[i];
53        }
54
55        if( bz_flag == 1 ){
56          /* bz_objid を ON にする為に I/O 通信プログラムにデータを送る(コマンドキュー登録) */
57          /* ※自動的に OFF を行なわせたい場合には、何か工夫してください                  */
58          kcxobj_sndistat_tokcx( bz_objid, (1) );
59        }
60
61        /* CPU を(このプログラムだけで独占(負荷)させない為に) 1 秒停止させる    */
62        /* ※sleep 関数に出会うと CPU は次の待ちプロセスに実行を移して行きます */
63        sleep( 1 );
64
65     }/*(while)*/
66
67 }/*(main)*/
```

[ライブラリ関数]

kcxobj_stat_ird() 関数 .... オブジェクトの状態値を取得

kcxobj_sndistat_tokcx() 関数 ..... オブジェクトへ設定値を出力 (コマンドキューにデータ登録)

3) kcxobj_stat_ird(objid, data)

int objid; ..... オブジェクトID

int *data; ..... オブジェクトの状態値(整数値)

4) kcxobj_sndistat_tokcx(objid, data)

int objid; ..... オブジェクトID

int data; ..... オブジェクトへの設定値(整数値)

以上でプログラムが完成しました。

5.7　自動監視制御プログラムのコンパイルとデバック

作成したプログラムを実行可能なプログラムにするためにコンパイルしてみましょう。「プログラム簡易編集」画面で“コンパイル”ボタンをクリックして下さい。コンパイル状況画面が表示されます。

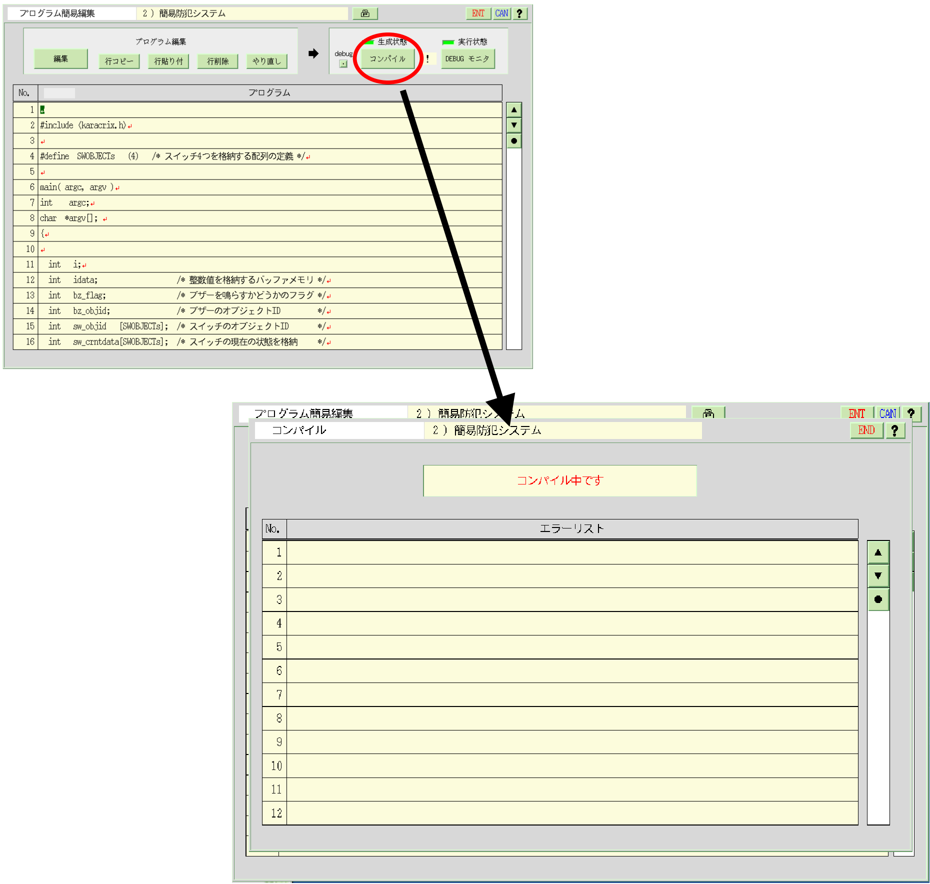


図 5.7.1 コンパイル状況画面

プログラムが、文法的に正しく記述してあれば、コンパイルが終了すると編集画面の生成状態インジケータが赤く点灯します。

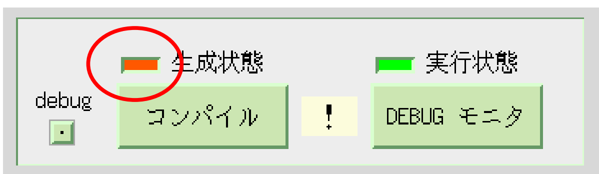


図 5.7.2 コンパイル成功時の生成状態インジケータ

もし、プログラムに文法的な間違いがあれば、以下の画面に示すように「コンパイル失敗」のメッセージが表示されます。

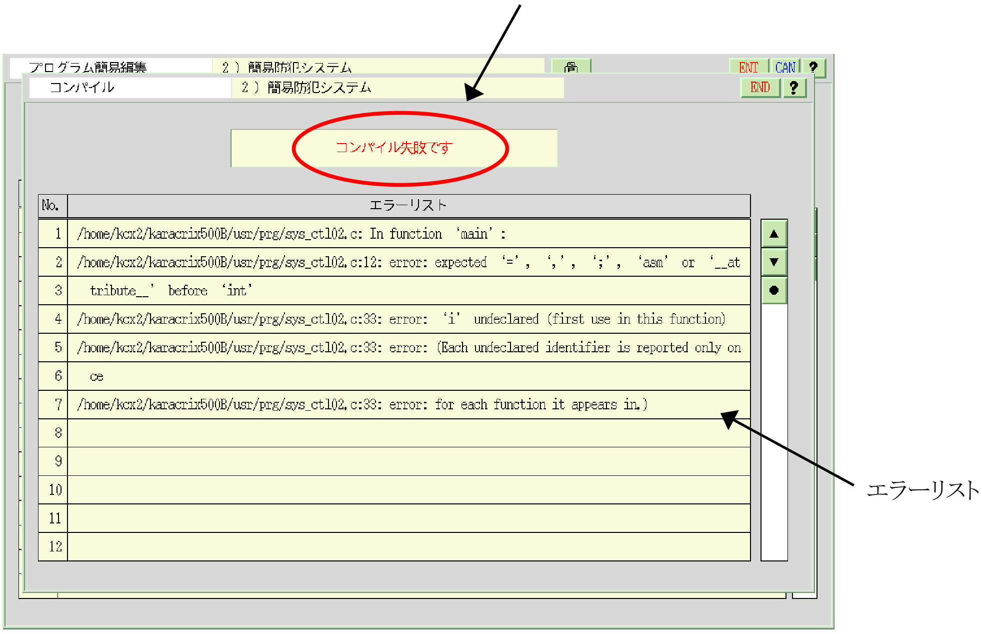


図 5.7.3 コンパイルエラー状況画面

この場合、エラーの箇所は、エラー行番号が表示されるなど何らかの手がかりが表示されますので、「プログラム簡易編集」画面で修正するか、またはプログラムを一旦外部へエクスポートしてエディタで編集した後に再度インポートするなどして、再びコンパイルを実行します。エラーメッセージが表示されなくなるまでプログラムをよく見直し修正して下さい。なお、当然ながらC言語の文法エラー以外の、プログラムフローなどのアルゴリズム・エラーは見つけられません。

※画面にエラーリストを表示しているのは、<u>OS付属の標準Cコンパイラ</u>です。

5.8　自動監視制御プログラムの実行

文法的に正しいプログラムが作成され、コンパイルも正常に終了して、実行ファイルが作成できたら、いよいよ実行してみます。「制御プログラム登録」画面に戻って下さい。

●プログラムを実行する

簡易防犯システムプログラムを実行する前に、「通信制御ドライバ　S1」が実行されていることを確認します。実行されていないときは実行しておきます。

“通信制御ドライバ(S1)”　が画面上に存在しない場合
「2章　通信制御ドライバ S1 について」→「2.4 通信制御ドライバ S1 のインストールとコンパイル」を参照して下さい。

簡易防犯システムプログラムの“実行”欄をクリックすると、実行ダイアログが表示されますので”RUN”ボタンを選択します[図 5.8.1]。正常に実行されると“実行”欄に”RUN”と表示されます。

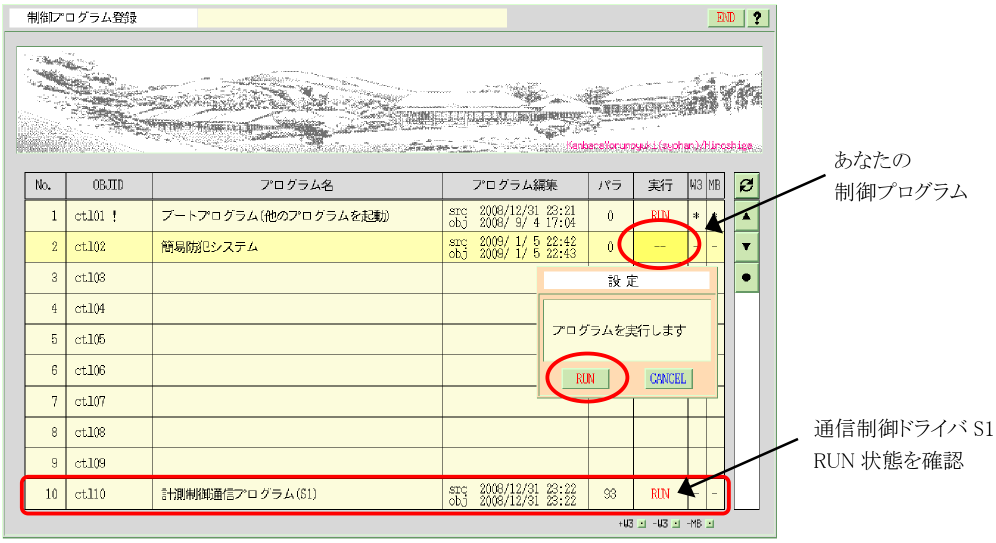


図 5.8.1　制御プログラムの実行

この状態で、窓の開閉などを行って見てください。 連動してブザーが鳴ればアルゴリズムの通りにうまく動いています。
ここで作成したプログラムでは、防犯スイッチ1～4のどれか一つがONになって、一度ブザーが警報を知らせたら鳴りっぱなしになるもっとも簡単なアルゴリズムを使っていますが、ある時間経たら自動的にブザーを停止させるとか、別のスイッチで一時停止させたり、色々と改良発展させることができると思います。

リスト 5.8.1　　簡易防犯システムプログラムリスト

```
#include <karacrix.h>

#define  SWOBJECTs   (4)   /* スイッチ 4 つを格納する配列の定義 */

main( int argc, char *argv[] )
{

  int   i;
  int   idata;                    /* 整数値を格納するバッファメモリ */
  int   bz_flag;                  /* ブザーを鳴らすかどうかのフラグ */
  int   bz_objid;                 /* ブザーのオブジェクト ID       */
  int   sw_objid   [SWOBJECTs];   /* スイッチのオブジェクト ID      */
  int   sw_crntdata[SWOBJECTs];   /* スイッチの現在の状態を格納     */
  int   sw_savedata[SWOBJECTs];   /* スイッチの前回の状態を格納     */

    /* KCX ライブラリの初期化(先頭の実行文に必須) */
    kcxinit( argc, argv );

    /* オブジェクトＩＤの取得 */
    sw_objid[0] = kcxobj_open( "di001" );  /* 防犯スイッチ 1 */
    sw_objid[1] = kcxobj_open( "di002" );  /* 防犯スイッチ 2 */
    sw_objid[2] = kcxobj_open( "di003" );  /* 防犯スイッチ 3 */
    sw_objid[3] = kcxobj_open( "di004" );  /* 防犯スイッチ 4 */
    bz_objid    = kcxobj_open( "do001" );  /* ブザー         */

    /* 初期設定 */
    /* 防犯スイッチの状態を検定する為の初期状態を取得しておく */
    for( i = 0; i < SWOBJECTs; i++ ){
      kcxobj_stat_ird( sw_objid[i], &idata );
      sw_savedata[i] = idata;
    }

    /* 防犯スイッチ 1～4 の状態を毎秒(下記 sleep(1)関数値依存)監視する永久ループ */
    while( 1 ){

      /* 防犯スイッチ 1～4 の現在の状態を取得 */
      for( i = 0; i < SWOBJECTs; i++ ){
         kcxobj_stat_ird( sw_objid[i], &sw_crntdata[i] );
      }

      /* 防犯スイッチ 1～4 のどこかで、ON になった時に限り         */
      /* つまり、前回の状態が 0 で、かつ現状態が 1 になった時に限り */
      /* ブザーフラグを ON とする                                   */

      bz_flag =  0;

      for( i = 0; i < SWOBJECTs; i++ ){
        if(( sw_crntdata[i] == (1) ) &&
           ( sw_savedata[i] == (0) ) ){
          bz_flag =  1;
        }
        sw_savedata[i] =  sw_crntdata[i];
      }

      if( bz_flag == 1 ){
        /* bz_objid を ON にする為に I/O 通信プログラムにデータを送る(コマンドキュー登録) */
        /* ※自動的に OFF を行なわせたい場合には、何か工夫してください                     */
        kcxobj_sndistat_tokcx( bz_objid, (1) );
      }

      /* CPU を(このプログラムだけで独占(負荷)させない為に) 1 秒停止させる    */
      /* ※sleep 関数に出会うと CPU は次の待ちプロセスに実行を移して行きます */
      sleep( 1 );

   }/*(while)*/

}/*(main)*/
```